Kony Fabric

SkySync Services

ORM API Guide

Release V8

Document Relevance and Accuracy

This document is considered relevant to the Release stated on this title page and the document version stated on the Revision History page. Remember to always view and download the latest document version relevant to the software release you are using.

September, 2017

Revision History

Date	Document Version	Description of Modifications/Release
09/08/2017	1.0	Document updated for release V8. Worked on rebranding of MobileFabric to Kony Fabric.

Table of Contents

1. Preface

Kony Sync Framework Sky ORM API enables developers to access Sky Engine eliminating developer overhead of writing complex Foreign Function Interface (FFI) calls. There are various APIs to help the developer meet their requirements. Platforms supported are iOS, Android, and Blackberry.

You can categorize these APIs as follows:

- Application Level
- Data Object / Scope Level
- Object Level

1.1 Purpose

This document enables developers to access Sky Engine eliminating developer overhead of writing complex Foreign Function Interface (FFI) calls.

1.2 Intended Audience

This document is intended for developers responsible for installing and configuring Kony Sky Sync Server.

1.3 Formatting Conventions

The following formatting conventions are used throughout the document:

Convention	Explanation
`Monospace`	■ User input text, system prompts, and responses ■ File path ■ Commands ■ Program code ■ File Names.
Italic	■ Emphasis ■ Names of books, and documents ■ New terminology.
Bold	■ Windows ■ Menus ■ Buttons ■ Icons ■ Fields ■ Tabs.
URL	Active link to a URL
Note	Provides helpful hints or additional information
Important	Highlights actions or information that might cause problems to systems or data

1.4 Contact Us

We welcome your feedback on our documentation. Write to us at techpubs@kony.com. For technical questions, suggestions, comments or to report problems on Kony product line, contact support@kony.com.

2. Pushing Changes to Sky MEAP Server

You can push changes to Sky MEAP Sever.

To push changes to Sky MEAP Server from device, you can call the following two APIs:

- skySync.startSession
- <dataObject>.dataObjectUpload

2.1 skySync.startSession

This API starts synchronization cycle and changes from server come to client and vice versa. You can call this API. Once this API is invoked, Sky Engine runs in the background and it continues to download and upload changes to Sky MEAP server. It continues to run till *skySync.stop* is called.

2.2 <dataObject>.dataObjectUpload

This API marks local changes to be uploaded to Sky MEAP server. The changes are uploaded immediately if the device is connected to Sky MEAP Server or uploaded later when the connectivity with the Sky MEAP server is established. You can invoke this API even when the device is offline. In general, if data in the local database is changed, the developer should call the *dataObjectUpload* method to push the changes to MEAP Server as indicated below.

```
function CreateProduct(){
     product = new Product();
     product.ProductName = "xyz";
     product.UnitPrice = 12.34;
     Product.create(successCallback, errorFailCallback);
}
function successCallback(){
       alert("Product Created Successfully");
     com.kony.SA.TRANSACTION.SA_TRANSACTION.dataObjectUpload
     (dataObjectUploadSuccess, dataObjectUploadError);
```

```
}
function errorFailCallback (res){
    alert("Product Creation Failed" + " with Error Code:" +
res.errorCode +
    ",error message:"+ res.errorMessage + ", error information:" +
    JSON.stringify(res.errorInfo));
}
function dataObjectUploadSuccess(){
          alert("Uploaded successfully");
}
 function dataObjectUploadError(res){
    alert("data Object upload failed" + " with Error Code:" +
res.errorCode +
    ",error message:"+ res.errorMessage + ", error information:" +
    JSON.stringify(res.errorInfo));
}
```

3. Application Level

This set of APIs help the developer to perform various ORM operations at Application level.

3.1 skySync.init

This API initializes Sky Sync engine. Internally, it calls *sky.provision* API. On successful, the connection establishes between device and Sky Sync engine. So you should call this function always after application re-start and before you perform any Sync operation.

3.1.1 Signature

skySync.init(config)

3.1.2 Configuration Parameters

Parameter Name [Mandatory/Optional]	Parameter Type	Description
SERVER [Mandatory]	String	The host name or IP address of SkyMobile gateway on which the target provisioning service runs. It may also be in the form of a URL.
PORT [Mandatory]	String	The port of SkyMobile gateway on which the target provisioning service is runs. Not required if a URL is used for the SERVER parameter.
PROFILE [Mandatory]	String	The name of SkyMobile provisioning profile to use when provisioning the device.
USEWIFI [Optional]	Boolean	Whether or not to use wifi for the connection. Permitted values are *TRUE*/*FALSE*. If no value is given *FALSE* is assumed.

Parameter Name [Mandatory/Optional]	Parameter Type	Description
CONNECTIONMODE (Optional/BlackBerry)	String	The connection mode you have to use when connecting as part of provisioning process. Permitted values are *BES*, *BIS-B*, *DIRECT*. If no value is given or any other value is given, *DIRECT* is assumed.
onProvisionError [Optional]	Function	Indicates the callback that is called in case the provision process encounters an error. It receives a context object with the below keys: 1. errorCode [String] - error Code 2. errorMessage [String] - Message 3. erroInfo [JSObject/null] - Further information about errors in various scopes
onProvisionSuccess [Optional]	Function	Indicates a callback that is called on successful provision.

3.1.3 Platform Availability

Available on all platforms mentioned.

3.1.4 Example

```
function syncInit()
{
	var config = {};
	//The below values should be set inorder to do provision.
		config.SERVER = "192.168.2.41";
		config.PORT = “60000”;
```

```
                config.PROFILE = "A"
                config.USEWIFI = true

        //onXXXXDemo is a function which will be called on these callbacks
                config.onProvisionError = onProvisionErrorDemo ;
                config.onProvisionSuccess = onProvisionSuccessDemo;



// The skySync.init command after populating the config object
                skySync.init(config);
}
```

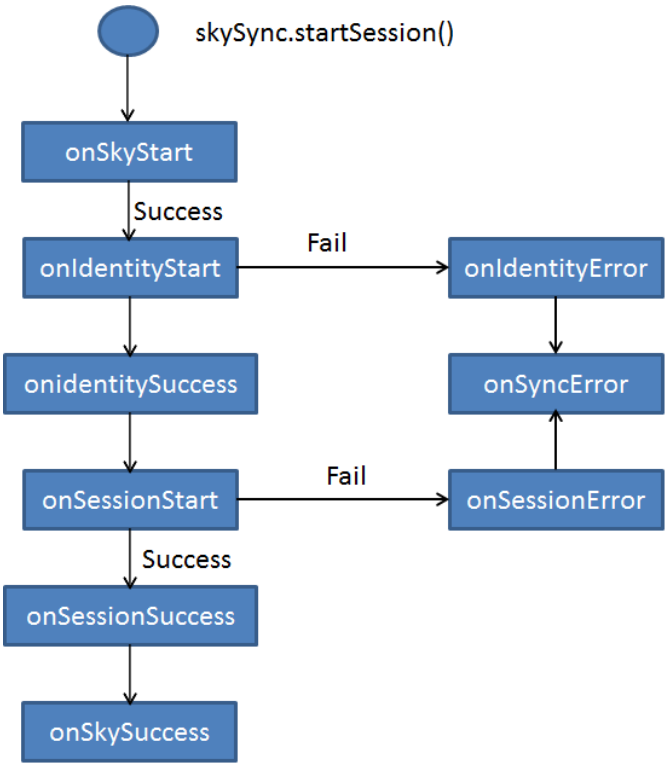
3.2 skySync.startSession

This function triggers the synchronization cycle. This function makes the device database ready with the following two steps:

1. identify: This method identifies the current user of SkySync engine, and leverages SkyMobile identity management processing.
2. start: This method starts Sky Sync engine. It immediately begins synchronizing data with the back-end Sky MEAP Server in background mode (asynchronously).

3.2.1 Client Side Lifecycle Events

The following image shows the client side lifecycle events:



The *skySync.startSession* receives a set of configuration as input that helps to fine tune the synchronization process to specific application Configuration Parameters.

3.2.2 Configuration Parameters

Parameter Name [Mandatory/Optional]	Parameter Type	Description
userid [Mandatory]	String	Specifies the user identifier to be used to authenticate the user to access the SKY Mobile gateway
password [Mandatory]	String	Specifies the password to be used to authenticate the user to access the SKY Mobile gateway
onSkyStart [Optional]	Function	Indicates the callback that is called before sky session starts

Parameter Name [Mandatory/Optional]	Parameter Type	Description
onSkySuccess [Optional]	Function	Indicates a final callback that is called after successful sky session
onSkyError [Optional]	Function	Indicates the final callback that is called for in case the sky start session process encounters an error. It receives a context object with the below keys: 1. errorCode [String] – error Code 2. errorMessage [String] - Message 3. erroInfo [JSObject/null] – Further information about errors in various scopes
onIndentifyStart [Optional]	Function	Indicates a callback that is called before sky identification starts
onIndentifyError [Optional]	Function	Indicates the final callback that is called for in case the sky identification process encounters an error. It receives a context object with the below keys: 1. errorCode [String] – error Code 2. errorMessage [String] - Message 3. erroInfo [JSObject/null] – Further information about errors in various scopes
onIndentifySuccess [Optional]	Function	Indicates a callback that is called after sky identification Successful
onSessionStart [Optional]	[Optional]	Indicates a callback that is called before sky start session starts

Parameter Name [Mandatory/Optional]	Parameter Type	Description
onSessionSuccess [Optional]	Function	Indicates a callback that is called before sky start session successful
onSessionError [Optional]	Function	Indicates the callback that is called for in case the sky start session process encounters an error. It receives a context object with the below keys: 1. errorCode [String] - error Code 2. errorMessage [String] - Message 3. erroInfo [JSObject/null] - Further information about errors in various scopes

3.2.3 Signature

sync.startSession(config)

3.2.4 Parameters

config [JSObject] - **Mandatory**

3.2.5 Platform Availability

Available on all platforms mentioned.

3.2.6 Example

```
function syncStart()
{
```

```
        var config = {};
        //The below values should be set inorder to do provision.
                config.USER = skyUserID;
                config.PASSWORD = skyPwd;

        //onXXXXCallback is a function which will be called on these
callbacks
                config.onSkyStart = onSkySyncStartCallback;
                config.onSkySuccess = onSkySyncSuccessCallback;
                config.onSkyError = onSkySyncErrorCallback;
                config.onIndentifyStart = onIndentifyStartCallback;
                config.onIndentifyError = onIndentifyErrorCallback;
                config.onIndentifySuccess = onIndentifySuccessCallback;
                config.onSessionStart = onSessionStartCallback;
                config.onSessionSuccess = onSessionSuccessCallback;
                config.onSessionError = onSessionErrorCallback;

        // The skySync.startSession command after populating the config
object
                skySync.startSession(config);
}
```

3.3 skySync.reset

This method resets SkySync engine back to its initial state (prior to provisioning). All user related data is destroyed, including the database, all logs and any associated binary files.

3.3.1 Signature

skySync.reset(config)

3.3.2 Parameters

config [JSObject] - **Mandatory**

3.3.3 Platform Availability

Available on all platforms mentioned.

3.3.4 Example

```
function resetSyncSkySession(){
	var config = {};
	config.onResetStart = syncSkyResetStartCallback ;
	config.onResetError = syncSkyResetErrorCallback ;
	config.onResetSuccess = syncSkyResetSuccessCallback;
	skySync.reset(config);
}
function syncSkyResetStartCallback(){
	alert("Sky Sync reset started");
}
function syncSkyResetSuccessCallback(){
	alert("Sky Sync reset successfull");
}
function syncSkyResetErrorCallback(outputparams){
	alert("Sky Sync reset error occured");
}
```

3.4 skySync.stop

This method stops SkySync engine. You should call this API if the application terminates, or the user logs out. The SkySync engine requires time to shut down gracefully.

3.4.1 Signature

skySync.stop(config)

3.4.2 Parameters

config [JSObject] - **Mandatory**

3.4.3 Platform Availability

Available on all platforms mentioned.

3.4.4 Example

```
function stopSyncSkySession(){
	var config = {};
	config.onStopStart =syncSkyStopStartCallback ;
	config.onStopError =syncSkyStopErrorCallback ;
	config.onStopSuccess = syncSkyStopSuccessCallback;
	skySync.stop(config);
}
function syncSkyStopStartCallback(outputparams){
	alert("Stopping Sky Sync sever started");
}
function syncSkyStopErrorCallback(outputparams){
	alert("Sky Sync stop error occured");
}
function syncSkyStopSuccessCallback(){
	alert("Sky Sync stop successful");
}
```

3.5 skySync.beginTransaction

This method begins a series of related updates, making them atomic. By default, all updates to SkySync data are committed to the database immediately. However, calling beginTransaction signals to the SkySync engine that the current thread is beginning a series of related updates that must be committed to the database as an atomic unit of work. Thereafter, all updates to the data made by the

calling thread are considered part of the same unit of work, until either commitTransaction or rollbackTransaction is called.

3.5.1 Signature

skySync.beginTransaction(config)

3.5.2 Parameters

config [JSObject] - **Mandatory**

3.5.3 Platform Availability

Available on all platforms mentioned.

3.5.4 Example

```
function beginTransaction(){
	var config = {};
	config.onSuccessTransaction =transactionSuccessCallback;
	skySync.beginTransaction(config);
}
function transactionStartedCallback(outputparams){
	alert("Transaction is started");
}
```

3.6 skySync.rollbackTransaction

This method performs an "undo" or rolls back a unit of work initiated by a call to beginTransaction, backing out all of the uncommitted changes made to the database by the current thread.

3.6.1 Signature

skySync.rollbackTransaction(config)

3.6.2 Parameters

config [JSObject] - **Mandatory**

3.6.3 Platform Availability

Available on all platforms mentioned.

3.6.4 Example

```
function rollbackTransaction(){
       var config = {};
       config.onSuccessTransaction =transactionRollbackCallback;
       skySync.rollbackTransaction(config)
}
function transactionRollbackCallback(outputparams){
       alert("Transaction is Rolled back");
}
```

3.7 skySync.commitTransaction

This method finalizes an atomic unit of work, writing it to the database. After you call the commitTransaction, any future updates to the data are again immediately written to the database, unless/until beginTransaction is called second time.

3.7.1 Signature

skySync.commitTransaction(config)

3.7.2 Parameters

config [JSObject] - **Mandatory**

3.7.3 Platform Availability

Available on all platforms mentioned.

3.7.4 Example

```
function rollbackTransaction(){
	var config = {};
	config.onSuccessTransaction =transactionRollbackCallback;
	skySync.rollbackTransaction(config)
}
function transactionRollbackCallback(outputparams){
	alert("Transaction is Rolled back");
}
```

3.8 Data Object / Scope Level

This set of APIs help the developer to perform various ORM operations at data object / scope level. Currently only one API is available in this category.

3.8.1 <dataObject>.dataObjectUpload

This uploads changes to MEAP server. It puts the changes in sky engine upload queue and background processes push the changes to server once device is online. You can invoke in offline / online mode.

3.8.1.1 Signature

<dataObject>.dataObjectUpload(successCallback, errorCallback)

3.8.1.2 Parameters

- successCallback**[function] - Optional**

Specifies the function that gets invoked on success

- errorCallback[function] - Optional

Specifies the function that gets invoked on error

3.8.1.3 Platform Availability

Available on all platforms mentioned.

3.8.1.4 Example

```
function dataObjectUpload(){
com.kony.SA.TRANSACTION.SA_TRANSACTION.dataObjectUpload
(dataObjectUploadSuccess,dataObjectUploadError);
 function dataObjectUploadSuccess(){
 alert("Uploaded successfully");
```

```
 }
 function dataObjectUploadError(res){
 alert("data Object upload failed" + " with Error Code:" +
res.errorCode +
",error message:"+ res.errorMessage + ", error information:" +
JSON.stringify(res.errorInfo));
 }
```

3.8.2 Data Object / Scope Level

This set of APIs help the developer to perform various ORM operations at data object / scope level. Currently only one API is available in this category.

3.8.2.1 <dataObject>.dataObjectUpload

This uploads changes to MEAP server. It puts the changes in sky engine upload queue and background processes push the changes to server once device is online. You can invoke in offline / online mode.

Signature

<dataObject>.dataObjectUpload(successCallback, errorCallback)

Parameters

- successCallback**[function] - Optional**

Specifies the function that gets invoked on success

- errorCallback[function] - Optional

Specifies the function that gets invoked on error

Platform Availability

Available on all platforms mentioned.

Example

```
function dataObjectUpload(){
com.kony.SA.TRANSACTION.SA_TRANSACTION.dataObjectUpload
(dataObjectUploadSuccess,dataObjectUploadError);
 function dataObjectUploadSuccess(){
 alert("Uploaded successfully");
 }
```

```
 function dataObjectUploadError(res){
 alert("data Object upload failed" + " with Error Code:" +
res.errorCode +
",error message:"+ res.errorMessage + ", error information:" +
JSON.stringify(res.errorInfo));
 }
```

4. Object Level

This set of APIs helps the developer to perform various ORM operations at a Sync Object level in the application.

To take advantage of object oriented programming in JavaScript, we gave some of the APIs in object oriented way. So, as far as JavaScript is concerned, we can classify ORMs as following:

1. Instance APIs:

   APIs that involve operations on single row can be invoked by standard JavaScript object as well as traditional way.

   Object Oriented Way

   For Example;

   ```
   var product = new Product();
   product.productId = 123
   product.name = "test";
   product. updateByPK (successcallback, error callback);
   ```

   Traditional Way

   For Example;

   ```
   Product. updateByPK ({productId :123}, { name:"test" },
   successcallback, error callback);
   ```

2. Static APIs:

   APIs that involve operations on multiple rows can only be invoked using traditional way.

   For Example;

```
Product. getAll ( successcallback, error callback);
```

The convention <object> below indicates the GlobalName of the SyncObject as specified in `Sync IDE / SyncConfig` file.

4.1 Various Configurations at Object Level

You can configure the following at object level. All parameters take boolean value *true*/*false*.

4.1.1 <object>.isCascadeInsertUpdateEnabled

If this parameter is *true*, creates ORM cascades insertion up the data object hierarchy until either a pre-existing record or the top of the hierarchy is reached. Internally, it calls *sky.dataObjectCascadeInsert*. If set to *false*, parent table is not be notified about insertion of child record. Internally, it calls *sky.tableInsert*.

4.1.2 <object>.isCascadeDeleteEnabled

If this parameter is *true*, deletebyPK/remove ORMs *cascades deletion through the data object. Internally, it calls sky.dataObjectCascadeDelete. If set to* false, internally, it calls *sky.tableDelete*.

4.1.3 <object>.isCascadeDeleteDown

This parameter determines whether to cascade downwards to the bottom of the data object hierarchy in deletebyPK/remove ORMs.

4.1.4 <object>.isDeleteIndicatorValueEnabled

This parameter determines value to write into the delete indicator field in deletebyPK/remove ORMs.

4.2 <object>.getAll (Static)

This API retrieves all instances of SyncObject present in local database.

4.2.1 Signature

<object>.getAll(successCallback, errorCallback, orderByMap)

4.2.1.1 Parameters

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

- orderByMap**[Array] - Optional**

  Specifies the way the data should be arranged in ascending/descending order of column name

4.2.1.2 Platform Availability

Available on all platforms mentioned.

4.2.1.3 Examples

General Example for *getAll*:

```
function syncGetAll(){
Account.getAll(showAccSuccessCallback,showAccFailCallback)
}

function showAccSuccessCallback(res){
    //res contains the data of all records of Account table
      alert("Get all Accounts success");
}

function showAccFailCallback(res){
alert("Get all Accounts failed" + " with Error Code:" +
```

```
res.errorCode + ",
error message:"+ res.errorMessage + ", error information:" +
JSON.stringify(res.errorInfo));
}
```

Example of orderByMap for *getAll*:

```
/* Define a orderByMap array to arrange the data in Categories table
in descending order of CategoryID and ascending order of
CategoryName */
 function CategoryGetAll(){
 var orderByMap = []
 orderByMap [0] = {};
 orderByMap [0].key = "CategoryID";
 orderByMap [0].sortType ="desc";
 orderByMap [1] = {};
 orderByMap [1].key = "CategoryName";
 orderByMap [1].sortType ="asc";
 Categories.getAll
(showCatSuccessCallback,showCatFailCallback,orderByMap);
}
function showCatSuccessCallback(res){
//res contains the data of Categories arranged in the way specified
alert("Get all Categories success");
//do something with the data contained in res
}
function showCatFailCallback(res){
 alert("Get all Accounts failed" + " with Error Code:" +
res.errorCode + ",
error message:"+ res.errorMessage + ", error information:" +
JSON.stringify(res.errorInfo));
}
```

4.3 <object>.getAllDetailsByPK (Instance)

This method retrieves a row using primary key from the local database.

4.3.1 Signature

<object>.getAllDetailsByPK(pk, successCallback, errorCallback)

4.3.1.1 Parameters

- pk**[number/JSObject]- Mandatory**

  Specify the primary key of the object using which the respective row data needs to be fetched from the database on the particular object. For composite primary keys it has to be a table/JSObject.

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.3.1.2 Platform Availability

Available on all platforms mentioned.

4.3.1.3 Example

```
//Static Way
function SyncGetAllDetailsByPK(){
       Product.getAllDetailsByPK
({ProductId:123},successCallback,errorFailCallback);
}
```

```
//OOP Way
function SyncGetAllDetailsByPK(){
product = new Product();
product.ProductId = 123;
product.getAllDetailsByPK(successCallback,errorFailCallback);
}
function successCallback(res){
    //res contains the details of product record specified by pk
alert("Get All Product Details By Primary Key Success");

   //do something with data contained in res
}

function errorFailCallback (res){
alert("Get All Details By Primary Key Failed + " with Error Code:" +
res.errorCode + ",
error message:"+ res.errorMessage + ", error information:" +
JSON.stringify(res.errorInfo));
}
```

4.4 <object>.create (Instance)

This API creates a new instance of sync object in the local Database.

4.4.1 Signature

<object>.create (object, succCallback, errorCallback)

4.4.1.1 Parameters

- Object**JSObject]- Mandatory**

  Specify the object that needs to be created in the database

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.4.1.2 Platform Availability

Available on all platforms mentioned.

4.4.1.3 Example

```
//Static Way
function CreateProduct(){
var objectProduct = {ProductName:"xyz", UnitPrice:12.34};

       Product.create(objectProduct, successCallback, errorFailCallback,
true);
}
//OOP Way
function CreateProduct(){
product = new Product();
product.ProductName = "xyz";
product.UnitPrice = 12.34;
/* This should be passed only when
Product.isCascadeInsertUpdateEnabled = true */
product.parentTable = {"USER_ID":"demo","OBJECT_TYPE":"ACTIV"};
Product.create(successCallback, errorFailCallback);
}

function successCallback(){
       alert("Product Created Successfully");
}
```

```
function errorFailCallback (res){
alert("Product Creation Failed" + " with Error Code:" +
res.errorCode + ",
error message:"+ res.errorMessage + ", error information:" +
JSON.stringify(res.errorInfo));
}
```

4.5 <object>.updateByPK(Instance)

This method updates sync object using primary key in the local Database.

4.5.1 Signature

<object>.updateByPK(pk, succCallback, errorCallback)

4.5.1.1 Parameters

- pk**[number/JSObject]- Mandatory**

  Specify the primary key object using which the respective row data needs to be updated in the database. For composite primary keys, it has to be a table/JSObject.

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.5.1.2 Platform Availability

Available on all platforms mentioned.

4.5.1.3 Example

```
//Static Way
function updateProduct(){
var objectProduct = {ProductName:"xyz", UnitPrice:12.34};
       Product.updateByPK(123, objectProduct, successCallback,
errorFailCallback, true);
}
// Object Oriented Way
function updateProduct (){
product = new Product();
product.ProductName = "xyz";
product.UnitPrice = 12.34;
product.ProductId = 123;
Product.updateByPK(successCallback, errorFailCallback);
}

function successCallback(){
       alert("Product updated Successfully");
}

function errorFailCallback (res){
       alert("Product updating failed" + " with Error Code:" +
res.errorCode + ",
error message:"+ res.errorMessage + ", error information:" +
JSON.stringify(res.errorInfo));
}
```

4.6 <object>.update(Static)

This method updates sync object using This API updates sync object using where clause in the local Database.

4.6.1 Signature

<object>.update (whereClause, succCallback, errorCallback)

4.6.1.1 Parameters

- whereclause**[Array]- Mandatory**

  Specify the array containing set of conditions specifying the rows to be selected

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.6.1.2 Platform Availability

Available on all platforms mentioned.

4.6.1.3 Example

```
function UpdateProduct(){
var objectProduct = {ProductName:"xyzUpdated", UnitPrice:15.34};
var conditionSet = [];
       conditionSet[ 0 ] = "UnitPrice > 10";

Product.update
(conditionSet,objectProduct,successCallback,errorFailCallback)
}
function successCallback(){
       alert("Update Product Success");
}
function errorFailCallback(res){
```

```
        alert("Update Product Failed" + " with Error Code:" + res.errorCode
+ ",

      error message:"+ res.errorMessage + ", error information:" +

      JSON.stringify(res.errorInfo));
}
```

4.7 <object>.deleteByPK (Instance)

This API deletes sync objects using primary key from the local Database.

4.7.1 Signature

<object>.deleteByPK(pk, succCallback, errorCallback)

4.7.1.1 Parameters

- pk**[number/JSObject] - Mandatory**

  Specify the array containing set of conditions specifying the rows to be selected

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.7.1.2 Platform Availability

Available on all platforms mentioned.

4.7.1.3 Example

```
//Static Way
function deleteProduct(){
Product. deleteByPK (123, successCallback, errorFailCallback, true);
}
//OOP Way
function deleteProduct (){
product = new Product();
product.ProductName = "xyz";
Product.deleteByPK (successCallback, errorFailCallback);
}

function successCallback(){
       alert("Product deleted Successfully");
}

function errorFailCallback(res){
       alert("Deleted Product By Primary Key Failed + " with Error Code:"
+
res.errorCode + ", error message:"+ res.errorMessage + ", error
information:"
+ JSON.stringify(res.errorInfo));
}
```

4.8 <object>.remove(Static)

This API deletes sync object(s) using where clause from the local Database. The record(s) are deleted from the enterprise datasource in the next Sync.

4.8.1 Signature

<object>.remove(whereclause, succCallback, errorCallback)

4.8.1.1 Parameters

- whereclause**[Array] - Mandatory**

  Specify the array containing set of conditions specifying the rows to be selected

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.8.1.2 Platform Availability

Available on all platforms mentioned.

4.8.1.3 Example

```
function DeleteProduct(){
      var conditionSet = [];
       conditionSet[ 0 ] = "ProductId > " + 100;
Product.remove(conditionSet, successCallback , errorFailCallback)
}

function successCallback(){
       alert("Product Deleted Successfully");
}

function errorFailCallback (res){
       alert("Product Delete Failed" + " with Error Code:" + res.errorCode
+ ",
error message:"+ res.errorMessage + ", error information:" +
JSON.stringify(res.errorInfo));
}
```

4.9 <object>.find(Static)

This API retrieves sync object(s) using where clause from the local Database.

4.9.1 Signature

<object>.find(wrcondition, succCallback, errorCallback)

4.9.1.1 Parameters

- wrconditon**[Array] - Mandatory**

  Specify the array containing set of conditions specifying the rows to be selected

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.9.1.2 Platform Availability

Available on all platforms mentioned.

4.9.1.3 Example

```
function loadEmployeeTerritories (){
var wcs = [];
wcs[0] = "OR"
wcs[1] = "EmployeeID > 30";
wcs[2] = "EmployeeName LIKE s%";
EmployeeTerritories.find(wcs, loadEmpTerritoriesSuccCallback,
loadEmpTerritoriesErrCallback);
}
```

```
function loadEmpTerritoriesSuccCallback(res){
       if((null != res && res.length> 0)){
               for ( i in res ){
                       //process the row
               }
       }
}

function loadEmpTerritoriesErrCallback(res)
{
alert("failed to retrieve records" + " with Error Code:" +
res.errorCode + ",
error message:"+ res.errorMessage + ", error information:" +
JSON.stringify(res.errorInfo));
}
```

4.10 <object>.getXXXwithYYY(Instance)

This API helps to retrieve all the records from the target object (XXX) corresponding to the foreign key attribute (YYY) primary key. In this case, <object> should have some relationship (One-To-Many, Many-To-One or One-To-One) with XXX using target attribute YYY. In case of Many-To-One, YYY are source attribute.

4.10.1 Signature

<object>.getXXXwithYYY(pk, succCallback, errorCallback)

4.10.1.1 Parameters

- pk**[Number/JSObject] - Mandatory**

  Specify the primary key of the object using which the respective row data needs to be fetched from the database

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.10.1.2 Platform Availability

Available on all platforms mentioned.

4.10.1.3 Example

```
//Static Way
function getOrderDetailsWithOrderID (){
Order.getOrderDetailsWithOrderID (123, successCallback,
errorFailCallback);
}
//OOP Way
function getOrderDetailsWithOrderID (){
order = new Order ();
order.orderId = 123;
order.getOrderDetailsWithOrderID(successCallback,
errorFailCallback);
}
function successCallback(res){
       //process the result
}
```

```
function errorFailCallback(res){
       alert("failed" + " with Error Code:" + res.errorCode + ", error
message:"+
res.errorMessage + ", error information:" + JSON.stringify
(res.errorInfo));
}
```

4.11 <object>.getCountOfXXXwithYYY(Instance)

This API helps to retrieve number of records from the target object (XXX) corresponding to the foreign key attribute (YYY) primary key. In this case, <object> should have some relationship (One-To-Many, Many-To-One or One-To-One) with XXX using target attribute YYY. In case of Many-To-One, YYY are source attribute.

4.11.1 Signature

<object>.getCountOfXXXwithYYY(pk, succCallback, errorCallback)

4.11.1.1 Parameters

- pk**[Number/JSObject] - Mandatory**

  Specify the primary key of the object using which the respective row data needs to be fetched from the database

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.11.1.2 Platform Availability

Available on all platforms mentioned.

4.11.1.3 Example

```
//Static Way
function getOrderDetailsWithOrderID (){
Order.getCountOfOrderDetailsWithOrderID (123, successCallback,
errorFailCallback);
}
//Object Oriented Way
function getOrderDetailsWithOrderID (){
order = new Order ();
order.orderId = 123;
order.getCountOfOrderDetailsWithOrderID(successCallback,
errorFailCallback);
}
function successCallback(res){
       alert("Count=" + res.count);
}

function errorFailCallback(res){
       alert("failed"+ " with Error Code:" + res.errorCode + ", error
message:"+
res.errorMessage + ", error information:" + JSON.stringify
(res.errorInfo));
}
```

4.12 <object>.removeDeviceInstancebyPK(Instance)

This API deletes sync objects using primary key from local Database. This does not have any effect in Sky MEAP in subsequent sync cycles.

4.12.1 Signature

<object>.removeDeviceInstancebyPK(pk, successcallback, errorcallback)

4.12.1.1 Parameters

- pk**[Number/JSObject] - Mandatory**

  Specify the primary key of the object using which the respective row data needs to be fetched from the database

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.12.1.2 Platform Availability

Available on all platforms mentioned.

4.12.1.3 Example

```
//Static Way
function removeDeviceInstancebyPK (){
Order. removeDeviceInstancebyPK (123, successCallback,
errorFailCallback);
}
Object Oriented Way
function removeDeviceInstancebyPK (){
order = new Order ();
order.orderId = 123;
order. removeDeviceInstancebyPK (successCallback,
errorFailCallback);
}
function errorFailCallback(res){
       alert("Removing Order Failed" + " with Error Code:" + res.errorCode
+ ",
```

```
error message:"+ res.errorMessage + ", error information:" +
JSON.stringify(res.errorInfo));
}
```

4.13 <object>.getAllCount(Static)

This API helps to retrieve total number of sync objects present in local database.

4.13.1 Signature

<object>.getAllCount (succCallback, errorCallback)

4.13.1.1 Parameters

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.13.1.2 Platform Availability

Available on all platforms mentioned.

4.13.1.3 Example

```
//Static Way
function getAllCount(){
Order.getAllCount(successCallback, errorFailCallback);
}
function successCallback(res){
       alert("Count=" + res.count);
}
```

```
function errorFailCallback(res){
       alert("failed"+ " with Error Code:" + res.errorCode + ", error
message:"+
res.errorMessage + ", error information:" + JSON.stringify
(res.errorInfo));
}
 + ", error information:" + JSON.stringify(res.errorInfo));
}
```

4.14 <object>.getCount(Static)

This API helps to retrieve total number of sync objects present in local database matching the where clause.

4.14.1 Signature

<object>.getCount (whereClause , succCallback, errorCallback)

4.14.1.1 Parameters

- whereClause**[Array] - Mandatory**

  Specify the array containing set of conditions specifying the rows to be counted

- successCallback**[function] - Optional**

  Specifies the function that gets invoked on success

- errorCallback**[function] - Optional**

  Specifies the function that gets invoked on error

4.14.1.2 Platform Availability

Available on all platforms mentioned.

4.14.1.3 Example

```
//Static Way
function getCount(){
var wcs = [];
wcs[1] = "OrderId > 230";
Order.getCount(wcs, successCallback, errorFailCallback);
}
function successCallback(res){
       alert("Count=" + res.count);
}

function errorFailCallback(res){
       alert("failed"+ " with Error Code:" + res.errorCode + ", error
message:"+
res.errorMessage + ", error information:" + JSON.stringify
(res.errorInfo));
}
```

5. Error Handling

Kony Sync Framework handles error handling through error codes. All the ORMs take errorCallback as a parameter that is called if any error occurs in that particular ORM. All ORMs return error parameter in errorCallback that is mapped and contains following keys:

1. **errorCode[number]**: Each error is assigned a specific error code
2. **errrorMessage[String]**: Description of the error scenario
3. **errorInfo[JSObject]**: Other information about error. It may also contain error intermediate errors also

Following table describes various error codes and corresponding error messages:

Error Code	Error Scenario	Error Message
-1001	ERROR_ GENERAL_ FAILURE	Error code to represent a general (unspecified) failure
-1002	ERROR_ ALREADY_ IDENTIFIED	Error code to indicate that the user is already identified
-1003	ERROR_ ALREADY_ PROVISIONED	Error code to indicate that the engine is already provisioned
-1004	ERROR_BAD_ CONDITION_SET	Error code to indicate that the condition set passed in as an argument was invalid
-1005	ERROR_ BLACKLISTED	Error code to indicate that the device is blacklisted

-1006	ERROR_ CONFIGURATION	Error code indicating a problem in the configuration data
-1007	ERROR_DATA_ OBJECT_NOT_ FOUND	Error code to indicate that the desired data object could not be found
-1008	ERROR_ DUPLICATE_ ITEM	Error code to indicate an attempt to insert a duplicate item (in other words, an attempt to create one that already exists)
-1009	ERROR_ ENCRYPTION_ HANDSHAKE	Error code indicating a problem with the encryption handshake
-1010	ERROR_ ENCRYPTION_ MISMATCH	Error code to indicate that there is a problem in the configuration of encryption parameters
-1011	ERROR_HOST_ IF_SETUP_ ERROR	Error code to indicate that there is a problem with the setup of the engine host interfaces. Simultaneous use of more than one back-end host is not permitted in conjunction with the SkySync engine
-1012	ERROR_HOST_ NOT_AVAILABLE	Error code to indicate that the host is not available (usually through lack of connectivity)
-1013	ERROR_ IDENTITY_ FAILURE	Error code indicating a general identity failure
-1014	ERROR_ INVALID_ PARAMETER	Error code to indicate that the given parameter was invalid

-1015	ERROR_IS_STARTED	Error code to indicate that the operation could not be completed, because the engine has been started
-1016	ERROR_MEAP_IS_DISABLED	Error code to indicate that the host MEAP is currently disabled
-1017	ERROR_NO_CONFIGURATION	Error code to indicate that no configuration data is available
-1018	ERROR_NO_INSTANCES_SELECTED	Error code to indicate that no profile instances suitable for use with the device could be found (selected)
-1019	ERROR_NOT_IDENTIFIED	Error code to indicate that the client has already identified
-1020	ERROR_NOT_PROVISIONED	Error code to indicate that the client is not provisioned
-1021	ERROR_NOT_RUNNING	Error code to indicate that the engine is not currently running, which in the majority of cases means it has not been started
-1022	ERROR_PROFILE_DEACTIVATED	Error code to indicate that the provisioning profile has been deactivated
-1023	ERROR_PROFILE_NOT_FOUND	Error code to indicate that the specified provisioning profile was not found
-1024	ERROR_RETRY	Error code to indicate that more interaction is required
-1025	ERROR_TABLE_NOT_FOUND	Error code to indicate that the desired table could not be found

-1026	ERROR_ UNCAUGHT_ EXCEPTION	Error code to indicate that an uncaught exception occurred
-1027	ERROR_USER_ NOT_FOUND	Error code to indicate that the user specified during provisioning was not found
7101	MISSING ATTRIBUTES FOR SKY	Missing attributes for Sky